A Symplectic Method for Approximating All the Eigenvalues of a Hamiltonian Matrix

C. Van Loan
Department of Computer Science
Cornell University
Ithaca, New York, 14853.

Submitted by Robert J. Plemmons

ABSTRACT

A fast method for computing all the eigenvalues of a Hamiltonian matrix M is given. The method relies on orthogonal symplectic similarity transformations which preserve structure and have desirable numerical properties. The algorithm requires about one-fourth the number of floating-point operations and one-half the space of the standard QR algorithm. The computed eigenvalues are shown to be the exact eigenvalues of a matrix $M + E$ where $\|E\|$ depends on the square root of the machine precision. The accuracy of a computed eigenvalue depends on both its condition and its magnitude, larger eigenvalues typically being more accurate.

1. MOTIVATION

A real $2n$-by-$2n$ matrix of the form

$$M = \begin{bmatrix} A^T & G \\ F & -A \end{bmatrix} \tag{1.1}$$

is called a Hamiltonian matrix if $A \in \mathbb{R}^{n\times n}$, $G^T = G \in \mathbb{R}^{n\times n}$, and $F^T = F \in \mathbb{R}^{n\times n}$. It is easy to verify that if

$$J = \begin{bmatrix} 0_n & I_n \\ -I_n & 0_n \end{bmatrix},$$

then

$$\mathcal{H} = \{ M \in \mathbb{R}^{2n\times 2n} \mid J^T M J = -M^T \}$$

52 Vanderbilt Ave., New York, NY 10017 0024-3795/84/$3.00

is precisely the set of all $2n$-by-$2n$ real Hamiltonian matrices. Here, I_n is the n-by-n identity.

For any matrix T let $\lambda(T)$ denote the set of its eigenvalues. It follows that

$$M \in \mathscr{H}, \quad \lambda \in \lambda(M) \quad \Rightarrow \quad -\lambda, \bar{\lambda}, -\bar{\lambda} \in \lambda(M). \tag{1.2}$$

This is because (1) M and $-M^T$ are similar and (2) the complex eigenvalues of a real matrix occur in conjugate pairs.

Hamiltonian matrices arise in several areas, including control theory. For example, in the simplest kind of linear-regulator problem we are asked to minimize

$$J(u) = \int_0^\infty \left[y^T y + u^T u \right] dt$$

subject to

$$\dot{x} = Ax + Bu, \qquad A \in \mathbb{R}^{n \times n}, \quad B \in \mathbb{R}^{n \times p},$$

$$y = Cx, \qquad C \in \mathbb{R}^{m \times n}$$

where $x(0) = x_0$ is given. It is widely known [3] that if this system is stabilizable and detectable and we set $F = BB^T$ and $G = C^T C$ in (1.1), then

(1) M has n eigenvalues $\lambda_1, \ldots, \lambda_n$ in the open left half plane,

(2) the unique symmetric positive definite solution P_* to the algebraic Riccati equation

$$0 = G + A^T P + PA - PFP$$

is given by $P_* = YZ^{-1}$ where $Y, Z \in \mathbb{R}^{n \times n}$ and the columns of $\begin{bmatrix} Y \\ Z \end{bmatrix}$ span M's invariant subspace associated with $-\lambda_1, \ldots, -\lambda_n$,

(3) the regulator problem is solved by $u_*(t) = -B^T P_* x(t)$, and

(4) $\lambda, \ldots, \lambda_n$ are the *closed-loop* eigenvalues, i.e., the eigenvalues of the optimally controlled system

$$\dot{x} = Ax - Bu_*(t) = (A - BB^T P_*) x(t).$$

This paper is motivated by the desire to find an efficient method for computing closed-loop eigenvalues.

A simple but inefficient approach to the Hamiltonian eigenvalue problem is the following:

ALGORITHM *MQR*.

Step 1. Form the $2n$-by-$2n$ array

$$M = \begin{bmatrix} A^T & G \\ F & -A \end{bmatrix}.$$

Step 2. Compute the Hessenberg reduction $U^T M U = H_M$, where U is orthogonal.

Step 3. Compute the eigenvalues of H_M by applying the unsymmetric QR algorithm.

The Hessenberg reduction and the QR algorithm are discussed in [6] and [8]. Algorithm *MQR* is particularly easy to implement—the user need only call the appropriate EISPACK subroutines. It requires approximately $64n^3$ floating-point operations (flops).

Unfortunately, M's rich Hamiltonian structure is ignored throughout *MQR*. This wastes computer time and storage. A way round these practical problems is to make use of symplectic matrices. $S \in \mathbb{R}^{2n \times 2n}$ is *symplectic* if

$$\begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix}^{-1} = \begin{bmatrix} S_{22}^T & -S_{12}^T \\ -S_{21}^T & S_{11}^T \end{bmatrix} \quad S_{ij} \in \mathbb{R}^{n \times n},$$

or, equivalently, if S belongs to the set

$$\mathcal{S} = \{ S \in \mathbb{R}^{2n \times 2n} \mid J^T S J = S^{-T} \}.$$

Note that $\mathcal{S}$ is closed under multiplication and that symplectic similarity transformations preserve Hamiltonian structure:

$$S \in \mathcal{S}, \qquad M \in \mathcal{H} \Rightarrow S^{-1} M S \in \mathcal{H}. \tag{1.3}$$

This follows because

$$J^T (S^{-1} M S) J = (J^T S J)^{-1} (J^T M J)(J^T S J) = -(S^{-1} M S)^T.$$

In this paper we present an algorithm for computing the eigenvalues of a Hamiltonian matrix that relies on orthogonal symplectic transformations. The

new method requires about one-fourth the number of floating-point operations needed by MQR and about half the storage. However, these positive attributes are partially offset by a somewhat less favorable error analysis.

The organization of the paper is as follows. In Section 2 we develop Householder symplectic and Givens symplectic transformations. We then show how these computational tools can be used to zero the (2, 1) block of the matrix M^2. Why this is a constructive course of action to take is shown in Section 3. The main algorithm and its computer implementation are detailed in Sections 4 and 5. In the final section we examine the numerical properties of the new method with a mixture of examples and analysis.

Readers already familiar with Householder and Givens transformations may find our exposition a bit lengthy, particularly if they have read [5]. However, we feel it is important to acquaint researchers in applied areas with the powerful tools of numerical algebra. For this reason our treatment is somewhat detailed.

2. ORTHOGONAL SYMPLECTIC MATRICES

If $Q \in \mathbb{R}^{2n\times 2n}$ is both orthogonal and symplectic, then the equation $J^TQJ = Q^{-T} = Q$ implies that

$$Q = \begin{bmatrix} Q_1 & Q_2 \\ -Q_2 & Q_1 \end{bmatrix}, \qquad Q_1, Q_2 \in \mathbb{R}^{n\times n}.$$

In this section we describe two types of orthogonal symplectic matrices that can be used to zero selected components of a vector.

First, there are the Householder symplectic matrices. These have the form

$$H(k, w) = \begin{bmatrix} \operatorname{diag}(I_{k-1}, P) & 0 \\ 0 & \operatorname{diag}(I_{k-1}, P) \end{bmatrix}$$

where

$$P = I - 2ww^T/w^Tw, \qquad w \in \mathbb{R}^{n-k+1}.$$

[We adopt the convention that if $w = 0$, then $H(k, w) = I_{2n}$.] Note that $H(k, w)$ is just a direct sum of two "ordinary" n-by-n Householder matrices. (See [8].)

Equally useful are the Givens symplectic matrices. These have the form

$$J(k,\theta)=\begin{bmatrix} C & S \\ -S & C \end{bmatrix}$$

where

$$C=\operatorname{diag}(I_{k-1},\cos\theta,I_{n-k})$$

$$S=\operatorname{diag}(0_{k-1},\sin\theta,0_{n-k}).$$

$J(k,\theta)$ is an "ordinary" $2n$-by-$2n$ Givens rotation that rotates in planes k and $k+n$. (See [8].)

As we mentioned, Householder and Givens symplectic transformations can be used to zero prescribed entries in a vector. In the Householder case we have

ALGORITHM *H*. Given k $(1\leqslant k<n)$ and $y,z\in\mathbb{R}^n$, the following algorithm determines $w=(w_k,\ldots,w_n)^T$ such that if

$$H(k,w)\begin{pmatrix} y \\ z \end{pmatrix}=\begin{pmatrix} v \\ x \end{pmatrix},$$

then $x_i=0$ for $i=k+1,\ldots,n$:

$$\sigma:=\left(z_k^2+\cdots+z_n^2\right)^{1/2}$$

$$w_k:=z_k+\operatorname{sign}(z_k)\,\sigma$$

$$w_i:=z_i \qquad \text{for} \quad i=k+1,\ldots,n$$

end

We point out that by interchanging the roles of y and z, Algorithm *H* can be used to determine $H(k,w)$ such that $v_i=0$ for $i=k+1,\ldots,n$.

While Householder symplectics can be used to zero large portions of a vector, Givens symplectics can be used to zero single entries:

ALGORITHM *J*. Given k $(1\leqslant k\leqslant n)$ and $y,z\in\mathbb{R}^n$, the following algorithm determines $c=\cos\theta$ and $s=\sin\theta$ such that if

$$J(k,\theta)\begin{pmatrix} y \\ z \end{pmatrix}=\begin{pmatrix} v \\ x \end{pmatrix},$$

then $x_k = 0$:

$$\begin{aligned}
&\sigma := \left(y_k^2 + z_k^2\right)^{1/2} \\
&\text{If } \sigma = 0 \\
&\quad \text{then } c := 1 \text{ and } s := 0 \\
&\quad \text{else } c := y_k/\sigma \text{ and } s := z_k/\sigma \\
&\text{end}
\end{aligned}$$

The above algorithms are organized for clarity. Careful computer implementations are considerably different and nontrivial. See the discussion of the BLAS (basic linear algebra subroutines) in the LINPACK user's manual [2].

3. THE SET OF SQUARED HAMILTONIAN MATRICES

We now outline how the eigenvalues of a $2n$-by-$2n$ Hamiltonian matrix M can be found using orthogonal symplectic similarity transformations. The key idea is to work with the matrix $N = M^2$. The case $n = 1$ suggests why the squaring of M leads to simplifications:

$$M^2 = \begin{pmatrix} a & g \\ f & -a \end{pmatrix}^2 = \operatorname{diag}\left(a^2 + fg, a^2 + fg\right).$$

Clearly, the eigenvalues of M are the two square roots of $a^2 + fg$.

For general n it turns out that squaring M effectively halves the dimension of the Hamiltonian eigenvalue problem. Before this can be shown it is necessary to establish some properties of squared Hamiltonians. Define the set $\mathcal{H}^2$ by

$$\mathcal{H}^2 = \left\{ N \in R^{2n \times 2n} \mid N = M^2, M \in \mathcal{H} \right\}.$$

Note that if

$$N = \begin{bmatrix} N_{11} & N_{12} \\ N_{21} & N_{22} \end{bmatrix} = \begin{bmatrix} A^T & G \\ F & -A \end{bmatrix}^2$$

where $F = F^T$ and $G = G^T$, then

$$\begin{aligned}
N_{22} &= A^2 + FG = N_{11}^T, \\
N_{12} &= A^T G - GA = -N_{12}^T, \\
N_{21} &= FA^T - AF = -N_{21}^T.
\end{aligned}$$

Thus, in a squared Hamiltonian matrix, the corner blocks are skew-symmetric and the diagonal blocks are transposes of each other.

$\mathscr{H}^2$ has two other exploitable properties. First, if $N \in \mathscr{H}^2$ and $S \in \mathscr{S}$, then $S^{-1}NS \in \mathscr{H}^2$, i.e., symplectic similarity preserves the property of being the square of a Hamiltonian matrix. This follows from (1.3) and the implication

$$N = M^2 \Rightarrow S^{-1}NS = (S^{-1}MS)^2.$$

Second, if $M \in \mathscr{H}$ and

$$\lambda(M) = \{\lambda_1, -\lambda_1, \ldots, \lambda_n, -\lambda_n\},$$

then

$$\lambda(M^2) = \{\lambda_1^2, \lambda_1^2, \ldots, \lambda_n^2, \lambda_n^2\}.$$

Thus, $\lambda(M)$ can be readily deduced once we compute $\mu_i = \lambda_i^2$ for $i = 1, \ldots, n$.

These observations form the basis of the following algorithm for computing the eigenvalues $\lambda_1, \ldots, \lambda_{2n}$ of $M \in \mathscr{H}$:

Step 1. Form

$$N = \begin{bmatrix} A^T & G \\ F & -A \end{bmatrix}^2.$$

Step 2. Compute an orthogonal symplectic Q such that

$$Q^T N Q = \begin{bmatrix} H & R \\ 0 & H^T \end{bmatrix},$$

where H is upper Hessenberg ($h_{ij} = 0$, $i > j + 1$).

Step 3. Use the QR algorithm to compute $\lambda(H) = \{\mu_1, \ldots, \mu_n\}$.

Step 4. For $i = 1, \ldots, n$ compute $\lambda_i = \sqrt{\mu_i}$, taking the square root located in the left half plane. Set $\lambda_{n+i} = -\lambda_i$ for $i = 1, \ldots, n$.

Step 2 is the only step in this process that requires immediate clarification. Details are given in the next section.

4. THE REDUCTION OF A SQUARED HAMILTONIAN

We now show how an orthogonal symplectic $Q \in \mathbb{R}^{2n\times 2n}$ can be determined such that

$$Q^T N Q = \begin{bmatrix} H & R \\ 0 & H^T \end{bmatrix}, \tag{4.1}$$

where $N \in \mathcal{H}^2$, and $H \in \mathbb{R}^{n\times n}$ is upper Hessenberg. It is worth illustrating a few steps of the algorithm before presenting it in full generality. We depict the original squared Hamiltonian matrix as follows:

$$N = \begin{bmatrix} D & U \\ V & D^T \end{bmatrix} = \begin{bmatrix} x & x & x & x & 0 & x & x & x \\ x & x & x & x & x & 0 & x & x \\ x & x & x & x & x & x & 0 & x \\ x & x & x & x & x & x & x & 0 \\ 0 & x & x & x & x & x & x & x \\ x & 0 & x & x & x & x & x & x \\ x & x & 0 & x & x & x & x & x \\ x & x & x & 0 & x & x & x & x \end{bmatrix} \qquad (n=4).$$

(The zero diagonals in U and V follow from skew symmetry.) It turns out that the (2,1) block of this matrix can be zeroed by applying a sequence of Householder and Givens symplectic similarity transformations.

The first step is to zero v_{31} and v_{41} using a Householder symplectic $H_1 = H(2, w)$. This matrix can be determined by executing Algorithm H with $k = 2$, $y = De_1$, and $z = Ve_1$. [Here, $e_1 = (1,0,0,0)^T$.] Notice that when a similarity transformation is performed with this matrix, only rows and columns 2, 3, and 4 of D, U, and V are affected, the result being

$$N = \begin{bmatrix} D & U \\ V & D^T \end{bmatrix} := H_1 N H_1^T = \begin{bmatrix} x & x & x & x & 0 & x & x & x \\ x & x & x & x & x & 0 & x & x \\ x & x & x & x & x & x & 0 & x \\ x & x & x & x & x & x & x & 0 \\ 0 & x & 0 & 0 & x & x & x & x \\ x & 0 & x & x & x & x & x & x \\ 0 & x & 0 & x & x & x & x & x \\ 0 & x & x & 0 & x & x & x & x \end{bmatrix}.$$

The zeros in the (1,3) and (1,4) positions of V follow because the updated N

is a squared Hamiltonian and thus has skew-symmetric corner blocks. This is why the zeros remain on the diagonals of U and V.

The next step is to zero v_{21} using a Givens symplectic similarity transformation. This can be achieved by applying Algorithm J with $k = 2$, $y = De_1$, and $z = Ve_1$. Let $J_1 = J(2, \theta)$ be the resulting transformation, and notice that only the second row and column of D, U, and V are affected by the update $J_1 N J_1^T$. This implies that

$$N = \begin{bmatrix} D & U \\ V & D^T \end{bmatrix} := J_1 N J_1^T = \begin{bmatrix} x & x & x & x & 0 & x & x & x \\ x & x & x & x & x & 0 & x & x \\ x & x & x & x & x & x & 0 & x \\ x & x & x & x & x & x & x & 0 \\ 0 & 0 & 0 & 0 & x & x & x & x \\ 0 & 0 & x & x & x & x & x & x \\ 0 & x & 0 & x & x & x & x & x \\ 0 & x & x & 0 & x & x & x & x \end{bmatrix}$$

Next, we compute a Householder symplectic to zero d_{31} and d_{41}. In particular, by applying Algorithm H with $k = 2$, $y = Ve_1$, and $z = De_1$ we obtain $G_1 = H(2, w)$ with the property that

$$N = \begin{bmatrix} D & U \\ V & D^T \end{bmatrix} := G_1 N G_1^T = \begin{bmatrix} x & x & x & x & 0 & x & x & x \\ x & x & x & x & x & 0 & x & x \\ 0 & x & x & x & x & x & 0 & x \\ 0 & x & x & x & x & x & x & 0 \\ 0 & 0 & 0 & 0 & x & x & 0 & 0 \\ 0 & 0 & x & x & x & x & x & x \\ 0 & x & 0 & x & x & x & x & x \\ 0 & x & x & 0 & x & x & x & x \end{bmatrix}$$

Note that only rows and columns 2, 3, and 4 of D, U, and V are altered by this similarity transformation.

This completes the zeroing in the first column of N. The computations we have illustrated, however, are typical of the general kth step. Overall we have the following "square-reduction" procedure:

ALGORITHM SR. Given

$$N = \begin{bmatrix} D & U \\ V & D^T \end{bmatrix} \in \mathcal{H}^2,$$

the following algorithm overwrites D with an upper Hessenberg matrix H having the property that $\lambda(N) = \lambda(H) \cup \lambda(H)$:

For $k = 1, \ldots, n-1$

If $k \leqslant n-2$ then apply Algorithm H with $y = De_k$ and $z = Ve_k$ to determine $H_k = H(k+1, w)$. Update as follows:

$$\begin{bmatrix} D & U \\ V & D^T \end{bmatrix} := H_k \begin{bmatrix} D & U \\ V & D^T \end{bmatrix} H_k^T.$$

Apply Algorithm J with $y = De_k$ and $z = Ve_k$ to determine $J_k = J(k+1, \theta)$. Update as follows:

$$\begin{bmatrix} D & U \\ V & D^T \end{bmatrix} := J_k \begin{bmatrix} D & U \\ V & D^T \end{bmatrix} J_k^T.$$

If $k \leqslant n-2$ then apply Algorithm H with $y = Ve_k$ and $z = De_k$ to determine $G_k = H(k+1, w)$. Update as follows:

$$\begin{bmatrix} D & U \\ V & D^T \end{bmatrix} := G_k \begin{bmatrix} D & U \\ V & D^T \end{bmatrix} G_k^T.$$

Upon completion, the array D contains the promised upper Hessenberg matrix.

The orthogonal matrix Q of (4.1) is given by

$$Q^T = J_{n-1}(G_{n-2}J_{n-2}H_{n-2}) \cdots (G_2J_2H_2)(G_1J_1H_1).$$

However, it is not necessary to accumulate this matrix if all that is required is the eigenvalues of M.

5. IMPLEMENTATION DETAILS

Care must be exercised when implementing Algorithm SR in order to avoid superfluous calculation and storage. We begin this section by highlighting some features of our software that reflect this concern.

The first point to discuss is the squaring of M. Normally, this matrix would be represented in two n-by-n arrays—one for A and one that is "shared" by the symmetric matrices F and G. (Of course, an n-vector is required to store

the diagonal of either F or G.) An equal amount of storage is necessary for N—an array for $A^2 + FG$ and an array for the skew-symmetric matrices $FA^T - AF$ and $A^TG - GA$. Unfortunately, a $3n^2/2$ workspace is required for the computation of N.

A way round this annoying problem is to work only implicitly with the matrix N in Algorithm SR by exploiting the equation

$$Q^TNQ = Q^TM^2Q = (Q^TMQ)^2.$$

Instead of applying the H_k, J_k, and G_k to N, we apply them to M. Of course, the construction of these orthogonal symplectic matrices requires access to certain components of Ne_k, the kth column of the current N. However, at any instant we can calculate these quantities from the current M via the formula $Ne_k = M(Me_k)$.

The resulting "implicit SR" algorithm is mathematically equivalent to the "explicit" version detailed in the previous section. In particular, it overwrites the original M with the Hamiltonian

$$M_0 = \begin{bmatrix} A_0^T & G_0 \\ F_0 & -A_0 \end{bmatrix} = Q^TMQ, \tag{5.1}$$

where Q is defined by (4.1). It follows that

$$A_0^{2^T} + G_0F_0 = H$$

is upper Hessenberg and that

$$F_0A_0^T - A_0F_0 = 0.$$

Hamiltonian matrices with the property that their square has a zero (2,1) block will be called *square-reduced*.

ALGORITHM SR (Implicit). Given

$$M = \begin{bmatrix} A^T & G \\ F & -A \end{bmatrix} \in \mathscr{H},$$

the following algorithm overwrites M with $M_0 = Q^TMQ$ where $Q \in \mathscr{S}$ is

orthogonal and M_0 is square-reduced:

For $k = 1,\ldots,n-1$
  If $k \leqslant n-2$ then
    Compute components $(k+1,\ldots,n)$ of $F(A^Te_k) - A(Fe_k)$.
    Compute $H_k = H(k, w)$ as in Algorithm SR.
    Update:

$$\begin{bmatrix} A^T & G \\ F & -A \end{bmatrix} := H_k \begin{bmatrix} A^T & G \\ F & -A \end{bmatrix} H_k^T.$$

  Compute the $k+1$st components of $A^T(A^Te_k) + G(Fe_k)$ and $F(A^Te_k) - A(Fe_k)$.
  Compute $J_k = J_k(k, \theta)$ as in Algorithm SR
    Update:

$$\begin{bmatrix} A^T & G \\ F & -A \end{bmatrix} := J_k \begin{bmatrix} A^T & G \\ F & -A \end{bmatrix} J_k^T.$$

  If $k \leqslant n-2$ then
    Compute components $(k+1,\ldots,n)$ of $A^T(A^Te_k) + G(Fe_k)$.
    Compute $G_k = H(k, w)$ as in Algorithm SR.
    Update:

$$\begin{bmatrix} A^T & G \\ F & -A \end{bmatrix} := G_k \begin{bmatrix} A^T & G \\ F & -A \end{bmatrix} G_k^T.$$

This algorithm requires $26n^3/3$ flops.

The efficient updating of M by an orthogonal symplectic similarity transformation requires some discussion. For Householder symplectics, if

$$H_k = H(k+1, w) = \operatorname{diag}(I_k, P, I_k, P)$$

and

$$M = \begin{bmatrix} A_{11}^T & A_{21}^T & G_{11} & G_{12} \\ A_{12}^T & A_{22}^T & G_{12}^T & G_{22} \\ F_{11} & F_{12} & -A_{11} & -A_{12} \\ F_{12}^T & F_{22} & -A_{21} & -A_{22} \\ k & n-k & k & n-k \end{bmatrix} \begin{matrix} k \\ n-k \\ k \\ n-k \end{matrix}$$

then

$$H_k M H_k^T = \begin{bmatrix} A_{11}^T & A_{21}^T P & G_{11} & G_{12} P \\ P A_{12} & P A_{22}^T P & P G_{12}^T & P G_{22} P \\ F_{11} & F_{12} P & -A_{11} & -A_{12} P \\ P F_{12}^T & P F_{22} P & -P A_{21} & -P A_{22} P \end{bmatrix}.$$

Thus, the update of M amounts to a collection of "ordinary" Householder updates. Computational details may be found in [8].

Updating by a Givens symplectic similarity $J_k = J(k, \theta)$ is equally simple. Set $c = \cos\theta$, $s = \sin\theta$, $w = A^T e_k$, $x = A e_k$, $y = G e_k$, and $z = F e_k$. Since $J_k M J_k^T$ affects just the kth row and column of A, F, and G, we need only perform the following calculations:

For $i = 1, \ldots, k-1$
- $a_{ik} := c x_i + s z_i$
- $a_{ki} := c w_i + s y_i$
- $f_{ki} := -s x_i + c z_i$
- $g_{ik} := c y_i - s w_i$

$a_{kk} := (c^2 - s^2) w_k + cs(y_k + z_k)$
$f_{kk} := c^2 z_k - s^2 y_k - 2cs w_k$
$g_{kk} := c^2 y_k - s^2 z_k - 2cs w_k$

For $i = k+1, \ldots, n$
- $a_{ik} := c x_i + s z_i$
- $a_{ki} := c w_i + s y_i$
- $f_{ik} := -s x_i + c z_i$
- $g_{ki} := c y_i - s w_i$

The lower triangular portion of the symmetric matrix F and the strictly upper triangular portion of the symmetric matrix G can be stored in a single n-by-n array. The diagonal of G can be stored in an n-vector. When updating these matrices it is only necessary to update the corresponding triangle.

Similar economies can be made when implementing the explicit SR algorithm. The work associated with both approaches to computing $\lambda(M)$ is as follows:

SR (explicit):

Formation of $N = M^2$	$4n^3$
Computing $Q^T N Q = \begin{bmatrix} H & R \\ 0 & H^T \end{bmatrix}$	$\frac{16}{3}n^3$
Computing $\lambda(H)$ via QR	$8n^3$
	$\frac{52}{3}n^3$

SR (implicit):

Computing $Q^TMQ = \begin{bmatrix} A_0^T & G_0 \\ F_0 & -A_0 \end{bmatrix}$ as in (5.1)	$\frac{26}{3}n^3$
Formation of $H = A_0^{2T} + G_0F_0$	n^3
Computing $\lambda(H)$ via QR	$8n^3$
	$\frac{53}{3}n^3$

Essentially, the two algorithms involve the same amount of work, about 25% of that required by MQR. It should be stressed, however, that the above style of quantifying work is only approximate.

Finally we mention that unlike the explicit algorithm, the implicit algorithm requires no more than $2n^2$ storage. The matrix $A_0^{2T} + G_0F_0$ can be formed as follows:

Overwrite F with the upper triangular portion of G_0F_0.
Store the subdiagonal portion of G_0F_0 in an n-vector w.
Accumulate the upper Hessenberg portion of A^{2T} in F and w.

6. NUMERICAL PROPERTIES AND EXAMPLES

Suppose $\hat{\lambda}_{MQR}$ is a computed eigenvalue of M obtained by using Algorithm MQR. If t-digit base-b floating-point arithmetic is used, then it can be shown that

$$\hat{\lambda}_{MQR} \in \lambda(M + E_{MQR}), \tag{6.1}$$

where $E_{MQR} \in \mathbb{R}^{2n\times 2n}$ satisfies

$$\|E_{MQR}\|_2 \approx b^{-t}\|M\|_2. \tag{6.2}$$

This says that $\hat{\lambda}_{MQR}$ is an exact eigenvalue of a matrix relatively "near" to M, an optimum result in that the mere storage of M results in rounding errors of order $b^{-t}\|M\|_2$.

In general, if $\hat{\lambda}_{MQR}$ is the computed analog of $\lambda \in \lambda(M)$ and λ is a simple eigenvalue, then

$$|\lambda - \hat{\lambda}_{MQR}| \approx \frac{b^{-t}\|M\|_2}{s(\lambda)} \tag{6.3}$$

where the quantity $1/s(\lambda)$ is the condition of λ. (The denominator $s(\lambda)$ is the cosine of the angle between the left and right eigenvectors associated with λ, a number that can be very small.) The results (6.1)–(6.3) follow immediately from the classical analysis in [8].

Although Algorithm *MQR* is "perfectly stable," it has one disconcerting numerical feature. Because it ignores Hamiltonian structure, the computed eigenvalues will not come in plus-minus pairs. Indeed, we have constructed examples where $n+1$ of the computed eigenvalues are situated in the open left half plane. This complicates the identification of the closed-loop eigenvalues in the regulator problem mentioned earlier.

Now let us consider the quality of the computed analog of λ when it is obtained by either implicit or explicit versions of Algorithm SR. Denoting this computed eigenvalue by $\hat{\lambda}_{SR}$, we will show that

$$\hat{\lambda}_{SR} \in \lambda(M + E_{SR}) \tag{6.4}$$

where $E_{SR} \in \mathbb{R}^{2n\times 2n}$ satisfies

$$\|E_{SR}\|_2 \approx \sqrt{b^{-t}}\|M\|_2. \tag{6.5}$$

Moreover, we will offer heuristic reasons why

$$|\hat{\lambda}_{SR} - \lambda| \approx \min\left\{ \frac{b^{-t}\|M\|_2^2}{s(\lambda)|\lambda|}, \frac{\sqrt{b^{-t}}\|M\|_2}{s(\lambda)} \right\}. \tag{6.6}$$

By comparing (6.3) with (6.6) we conclude that

(1) If $\|M\|/|\lambda| \approx 1$, then $\hat{\lambda}_{SR}$ is essentially as accurate as $\hat{\lambda}_{MQR}$.

(2) If $\|M\|/|\lambda|$ is large, then the error in $\hat{\lambda}_{SR}$ may be up to $\sqrt{b^t}$ times as large as the error in $\hat{\lambda}_{MQR}$. Similar comments apply if λ is not a simple eigenvalue.

We have run numerous examples on Cornell's IBM 370 computer ($b^{-t} = 16^{-14} \approx 10^{-17}$) which confirm those contentions. The following are typical:

EXAMPLE 1. M equals the 18-by-18 Hamiltonian arising from the high-speed-vehicle control problem described in [4]. The eigenvalues are reasonable-sized and well conditioned. For all 18 eigenvalues, $|\hat{\lambda}_{SR} - \hat{\lambda}_{MQR}| \approx 10^{-14}$.

EXAMPLE 2. $M = Q'\text{diag}(D, -D)Q^T$, where $D = \text{diag}(1, 10^{-2}, 10^{-4}, 10^{-6}, 10^{-8})$ and Q is a randomly generated orthogonal symplectic matrix. Because M is symmetric, the eigenvalues are perfectly well conditioned, i.e., for each eigenvalue, $s(\lambda) = 1$. For stable eigenvalues we have

λ	$\lvert\hat{\lambda}_{SR} - \lambda\rvert$	$\lvert\hat{\lambda}_{MQR} - \lambda\rvert$
-1	10^{-15}	10^{-15}
-10^{-2}	10^{-15}	10^{-16}
-10^{-4}	10^{-13}	10^{-17}
-10^{-6}	10^{-12}	10^{-16}
-10^{-8}	10^{-9}	10^{-17}

As predicted by (6.6), the accuracy of $\hat{\lambda}_{SR}$ diminishes with its magnitude.

EXAMPLE 3. $M = Q\text{diag}(H, -H)Q^T$, where Q is a randomly generated orthogonal symplectic matrix and H is the 12-by-12 Frank matrix. (See [7].) M has some very ill-conditioned eigenvalues. For the four worst-conditioned stable eigenvalues we have

$\lambda \approx$	$s(\lambda) \approx$	$\lvert\hat{\lambda}_{SR} - \hat{\lambda}_{MQR}\rvert$
$-.1436$	10^{-7}	10^{-7}
$-.0812$	10^{-8}	10^{-6}
$-.0495$	10^{-8}	10^{-6}
$-.0310$	10^{-8}	10^{-7}

Thus, even if an eigenvalue is not particularly small, $\hat{\lambda}_{SR}$ can differ significantly from $\hat{\lambda}_{MQR}$ if λ is ill conditioned.

EXAMPLE 4. Several $n = 50$ random examples were run so as to compare the actual execution times of the implicit SR algorithm with MQR. Not surprisingly, the speedup was a little less than what the operation counts predicted. This is no doubt due to the more complicated looping and index checking in SR. The experiments suggested that the SR execution time is about 30% of that for MQR.

In the remainder of this section we justify (6.4)–(6.6). To do this we must establish the following theorem concerned with the singular values of a shifted Hamiltonian. (Readers unfamiliar with singular values should consult [6].)

THEOREM 6.1. *Let M be a real $2n$-by-$2n$ Hamiltonian matrix. If $\lambda \in \mathbb{C}$, then the matrices $M - \lambda I$, $M + \lambda I$, $M - \bar{\lambda} I$, and $M + \bar{\lambda} I$ have the same singular values.*

Proof. Let

$$U^H(M - \lambda I)V = \Sigma = \operatorname{diag}(\sigma_i) \tag{6.7}$$

be the singular-value decomposition (SVD) of $M - \lambda I$ with

$$U^H U = V^H V = I_{2n}$$

and

$$\sigma_1 \geqslant \cdots \geqslant \sigma_{2n} \geqslant 0.$$

For any matrix $W = (w_{ij})$ let $\overline{W} = (\overline{w}_{ij})$ denote its conjugate. It then follows from (6.7) and $J^T M J = -M^T$ that

$$\begin{aligned} (-JV)^H(M + \bar{\lambda} I)(JU) &= \Sigma, \\ \overline{U}^H(M - \bar{\lambda} I)\overline{V} &= \Sigma, \\ (-J\overline{V})^H(M + \lambda I)(J\overline{U}) &= \Sigma. \end{aligned}$$

Since $\overline{U}$, $\overline{V}$, and J are each unitary, these are the SVDs of $M + \bar{\lambda} I$, $M - \bar{\lambda} I$, and $M + \lambda I$ respectively. ∎

We now have the tools to establish (6.5). It can be shown that

$$\hat{\lambda}_{\mathrm{SR}}^2 \in \lambda(M^2 + E) \tag{6.8}$$

where $E \in \mathbb{R}^{2n \times 2n}$ and $\|E\|_2 \approx b^{-t}\|M\|_2^2$. This follows by standard inverse error-analysis techniques, details of which are given in [9]. It follows that a unit 2-norm vector $x \in \mathbb{C}^{2n \times 2n}$ exists such that

$$0 = \left(M^2 - \hat{\lambda}_{\mathrm{SR}}^2 I + E\right)x = (M - \hat{\lambda}_{\mathrm{SR}} I)(M + \hat{\lambda}_{\mathrm{SR}} I)x + Ex$$

Thus,

$$\|E\|_2 \geqslant \|Ex\|_2 = \|(M - \hat{\lambda}_{\mathrm{SR}} I)(M + \hat{\lambda}_{\mathrm{SR}} I)x\|_2 \tag{6.9}$$

Let $\sigma_{\min}(W)$ denote the smallest singular value of a square matrix W. It is well known that

$$\|Wx\|_2 \geqslant \sigma_{\min}(W)\|x\|_2$$

for any vector x. Moreover,

$$\sigma_{\min}(W) = \min_{\det(W+E)=0} \|E\|_2.$$

Consequently, using (6.9) and the theorem, we have

$$\begin{aligned}\|E\|_2 &\geq \sigma_{\min}(M - \hat{\lambda}_{SR} I)\sigma_{\min}(M + \hat{\lambda}_{SR} I) \\ &= [\sigma_{\min}(M - \hat{\lambda}_{SR} I)]^2.\end{aligned}$$

It follows that there is a matrix E_{SR} (possibly complex) satisfying

$$\|E_{SR}\|_2 = \sigma_{\min}(M - \hat{\lambda}_{SR} I) \leq \sqrt{\|E\|_2} \approx \sqrt{b^{-t}}\|M\|$$

such that $M - \hat{\lambda}_{SR} I + E_{SR}$ is singular. This establishes (6.4) and (6.5).

If $\hat{\lambda}_{SR}$ is the computed analog of λ, then from standard eigenvalue perturbation theory we have

$$|\lambda - \hat{\lambda}_{SR}| \approx \frac{\sqrt{b^{-t}}\|M\|}{s(\lambda)} \tag{6.10}$$

where $1/s(\lambda)$ is the condition of λ. On the other hand, it follows from (6.8) that

$$|\lambda^2 - \hat{\lambda}_{SR}^2| \approx \frac{b^{-t}\|M\|^2}{s(\lambda^2)},$$

where $1/s(\lambda^2)$ is the condition of λ^2 as an eigenvalue of M^2. It can be shown that $s(\lambda) = s(\lambda^2)$, and since $(\lambda^2 - \hat{\lambda}_{SR}^2) = (\lambda - \hat{\lambda}_{SR})(\lambda + \hat{\lambda}_{SR})$, we have approximately

$$|\lambda - \hat{\lambda}_{SR}| \approx \frac{b^{-t}\|M\|^2}{s(\lambda)|\lambda|}.$$

This heuristic equation coupled with (6.10) justifies (6.6).

7. CONCLUSIONS

We have presented a fast algorithm for approximating all the eigenvalues of a Hamiltonian matrix. The method exploits structure and requires a

minimum of storage. It is preferable to the ordinary Q-R approach except in those problems where the Hamiltonian M has a small eigenvalue ($|\lambda| \leqslant \sqrt{b^{-t}}\|M\|_2$) *and* it is necessary to compute this eigenvalue as accurately as possible. We suspect that this will rarely be necessary in most control engineering applications. (One typically does not design systems that have eigenvalues that are so dangerously close to being unstable as $\lambda = -\sqrt{b^{-t}}$.)

Finally, we mention that the techniques described in this paper are currently being used to develop "symplectic methods" for the algebraic and differential Riccati equations.

I wish to thank Ralph Byers for performing the numerical experiments and for rigorously establishing Equation (6.8). I am also grateful to Ian Gladwell and the referee for reading the original manuscript and suggesting improvements.

This paper was produced while I was a visitor in the Department of Mathematics at the University of Manchester, England. Financial support for my visit was provided by the Science Research Council (Grant number GR/B/95257) and the National Science Foundation (Grant MCS80-04106).

REFERENCES

1 B. T. Smith, J. M. Boyle, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler, *Matrix Eigensystem Routines*—EISPACK *Guide*, 2nd ed., Springer, New York, 1970.

2 J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, LINPACK *Users Guide*, SIAM, Philadelphia, 1979.

3 H. Kwackernaak and R. Sivan, *Linear Optimal Control Systems*, Wiley-Interscience, New York, 1972.

4 A. J. Laub, A Schur method for solving algebraic Riccati equations, *IEEE Trans. Automat. Control* AC-24:913–921 (1979).

5 C. C. Paige and C. Van Loan, A Hamiltonian-Schur Decomposition, *Linear Algebra Appl.*, to appear.

6 G. W. Stewart, *Introduction to Matrix Computations*, Academic, New York, 1972.

7 J. H. Wilkinson, *Rounding Errors in Algebraic Processes*, Prentice-Hall, Englewood Cliffs, N.J., 1963.

8 J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford U.P., Oxford, 1965.

9 R. Byers, Algorithms for Hamiltonian and symplectic eigenproblems, Ph.D. Thesis, Field of Applied Mathematics, Cornell Univ., Ithaca, N.Y., 1982.

Received 16 November 1982; revised June 1983